

TV CHECKER



Instituto Politécnico de Beja
Tecnologias Web e Dispositivos Móveis
Programação de Aplicação do lado do Cliente

15035 – Joana Santinhos
18823 – Guilherme Doutor
18825 – João Caeiro
18829 – Patrícia Costa

ÍNDICE

TV CHECKER.....	1
Introdução.....	4
Descrição das fases	5
Primeira Fase – Organização do Trabalho.....	5
Segunda Fase – Definição da proposta de trabalho.....	5
Terceira Fase – Base de Dados	5
Quarta Fase - Funcionalidades	5
Quinta Fase – Protótipo	5
Primeira Fase - Organização do Trabalho	6
Implementação do GitHub.....	6
Definição dos meios de comunicação entre os elementos do grupo	6
Relatório	6
Ferramenta de organização de tarefas	6
Escolha do líder para as tarefas	6
Segunda Fase – Definição da proposta de trabalho.....	7
Desenhos da aplicação	7
Protótipos de baixa definição.....	7
Protótipos de alta definição	10
Identificação de um sistema/aplicação inspiradora	12
Caracterização dos utilizadores.....	12
Cenário de utilização	13
Análise de Tarefas	14
Terceira fase – Base de Dados.....	17
Quarta Fase – Implementação do código	18
Ecrãs Principais da aplicação	18
Ecrã de Login	18
Ecrã de Signup	19
Ecrã da Página Inicial.....	21
Implementação da BD	22
Website de divulgação	24
HTML e CSS.....	26
Conclusões	28

Índice de Figuras

Ilustração 1 Diagrama de navegação	7
Ilustração 2 Desenho da aplicação mobile.....	8
Ilustração 3 Desenho do sítio web	9
Ilustração 4 Printscreen do Figma.....	10
Ilustração 5 Printscreen do Ai	11
Ilustração 6 Printscreens da app TV Time	12
Ilustração 7 Modelo Relacional da BD	17
Ilustração 8 Ecrã inicial.....	18
Ilustração 9 Código da MainActivity.....	18
Ilustração 10 Ecrã Sign Up	19
Ilustração 11 Código da SignUpActivity	20
Ilustração 12 Ecrã Home Page.....	21
Ilustração 13 Código da HomePageActivity	21
Ilustração 14 Código da AppDatabase	22
Ilustração 15 DAOs da classe User	23
Ilustração 16 DAOs da classe Series	23
Ilustração 17 DAOs da classe Categorias.....	23
Ilustração 18 Código do build.gradle (Glide e Room)	24
Ilustração 19 HTML	24
Ilustração 20 CSS	25
Ilustração 21 Design no Ai.....	25
Ilustração 22 Website implementado.....	26
Ilustração 23 Website implementado visão telemóvel.....	26
Ilustração 24 CSS adaptado a ecrãs.....	27

Introdução

Este trabalho foi projetado no âmbito da disciplina “Programação de Aplicação do Lado do Cliente”, no curso Tecnologias Web e Dispositivos Móveis, no IPBeja. Neste projeto, elaborou-se uma aplicação Android com acesso à base de dados, cujo tema foi escolhido pelos alunos. Escolheu-se uma aplicação, “TV Checker”, onde os utilizadores pudessem guardar todos os episódios das séries que vêm ou viram, colocar séries nos favoritos, e até anotar os minutos em que ficaram num certo episódio.

Este trabalho tem como principal objetivo enriquecer os conhecimentos dos alunos, realizando uma aplicação para telemóvel que tenha várias funcionalidades e que seja viável em termos de mercado.

Neste relatório faz-se referência a todas as fases do trabalho, procedendo à explicação das mesmas.

Descrição das fases

Primeira Fase – Organização do Trabalho

1. Implementação do Github
2. Definição dos meios de comunicação entre os elementos do grupo
3. Divisão de tarefas
4. Escolha do Líder

Segunda Fase – Definição da proposta de trabalho

5. Desenho da aplicação e respetivo sítio web
6. Identificação de um sistema inspirador
7. Criação de vários utilizadores com determinadas características cada

Terceira Fase – Base de Dados

8. Criação da Base de Dados
9. Teste de funcionalidade da mesma

Quarta Fase - Funcionalidades

10. Funcionalidades a implementar na app/website

Quinta Fase – Protótipo

11. Avaliação do Protótipo

Primeira Fase - Organização do Trabalho

Implementação do GitHub

Na Implementação do GitHub, foi nos dada a conhecer a aplicação por sugestão dos professores, para que todos os membros do grupo pudessem partilhar e aceder a todos os ficheiros do grupo.

Definição dos meios de comunicação entre os elementos do grupo

Os meios de comunicação mais utilizados para a comunicação de grupo foi o “Discord” e o “Messenger”. O “Discord” foi escolhido por ser uma ferramenta que permite afixar mensagens, enviar vários tipos de ficheiros, fazer partilhas de ecrã, e chamadas de voz com todos os membros. O “Messenger” foi também algumas vezes utilizados para serem enviadas, rapidamente, fotografias relativas ao trabalho e *printscreens*.

Relatório

O relatório, em grande parte, foi feito na plataforma online "Pages", que pertence ao iCloud da Apple, onde todos podem trabalhar ao mesmo tempo no mesmo documento. Algumas funcionalidades não era suportadas pelo “Pages”, e nessa altura utilizou-se o Word para realizar as configurações finais.

Ferramenta de organização de tarefas

De início a aplicação que foi utilizada na organização de tarefas foi o "Trello", onde foi possível organizar melhor as tarefas que foram realizadas, as que estavam a ser realizadas e as que ainda não tinham sido inicializadas.

Escolha do líder para as tarefas

A escolha dos líderes para as tarefas designadas foi feita pela preferência de cada um, conhecimentos e facilidade de desempenho. Tendo sido feita da seguinte forma:

- Desenho da aplicação e sítio web, **Joana Santinhos**
- Base de dados, **João Caeiro**
- Código, **Patrícia Costa**
- Relatório e apresentação, **Guilherme Doutor**

Segunda Fase – Definição da proposta de trabalho

Desenhos da aplicação

Protótipos de baixa definição

Inicialmente, desenhou-se em papel o desenho da aplicação e do sítio web. Nessa altura definiu-se também a navegação da aplicação(figura 1).

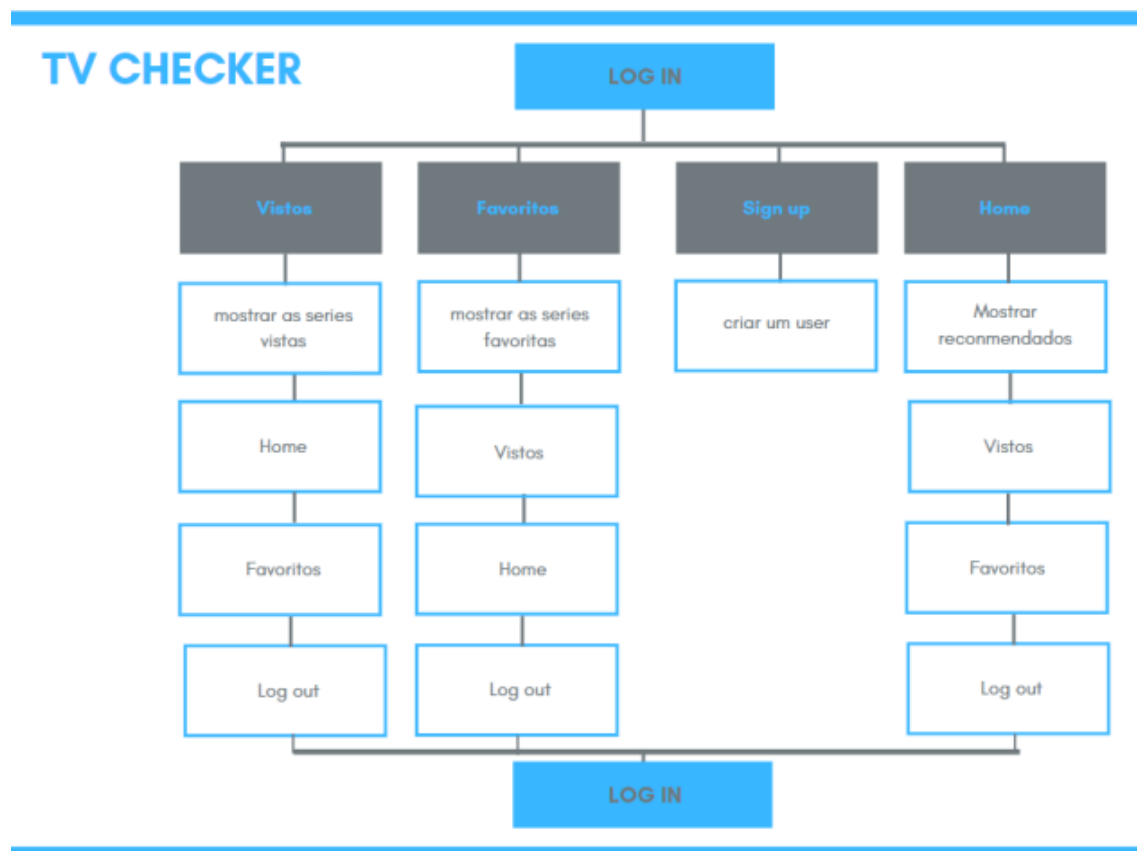


Ilustração 1 Diagrama de navegação

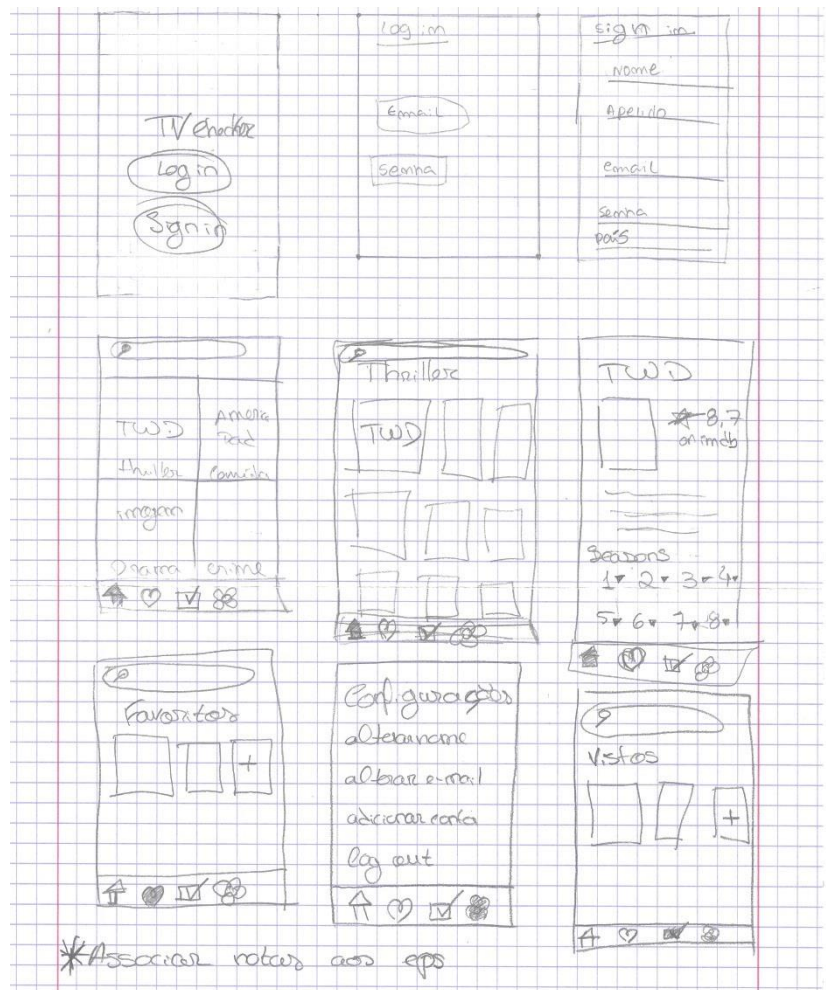


Ilustração 2 Desenho da aplicação mobile

A página inicial tem o *login* e o *signup*. Ao entrar, o utilizador tem a página principal, onde se encontra as categorias. Após entrar na *app*, existe sempre um menu presente, onde o utilizador sabe sempre onde está e pode rapidamente passar para outro ecrã. No menu, encontra-se o botão da página inicial, dos favoritos, dos vistos, e as definições. (Figura 1)

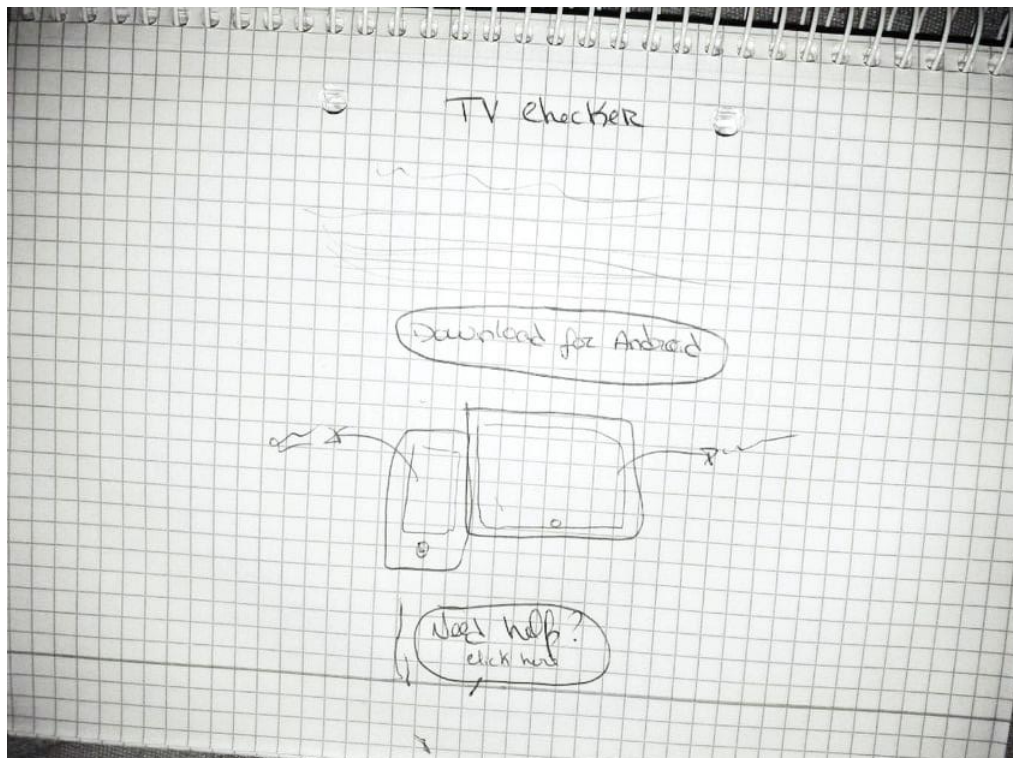


Ilustração 3 Desenho do sítio web

Para o *website*, desenhou-se uma página onde consta o logótipo, um pequeno texto explicativo da *app*, um botão para *download*, e ecrãs da *app* com as respetivas explicações.(Figura 2)

Protótipos de alta definição

Após os desenhos, fizeram-se protótipos com maior qualidade e detalhe. O protótipo da aplicação foi desenhado no “Figma”, no *browser* (figura 3), e o do sítio *web* foi desenhado no Adobe Illustrator (figura 4).

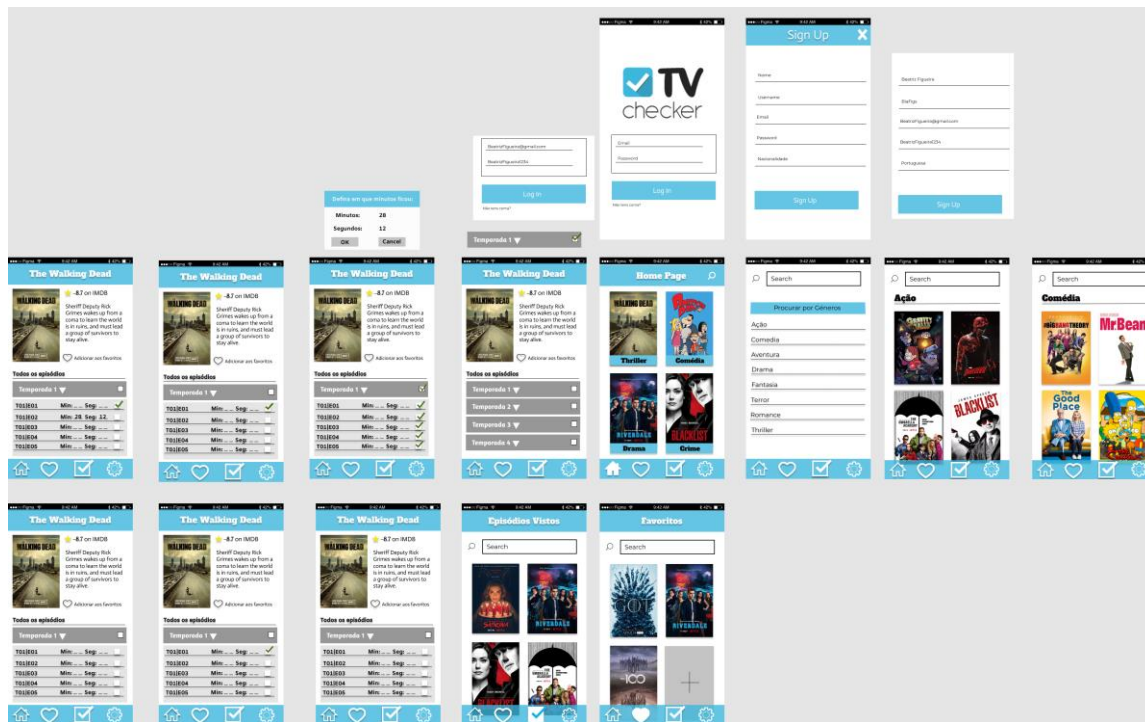


Ilustração 4 Printscreen do Figma



Ilustração 5 Printscreen do Ai

Identificação de um sistema/aplicação inspiradora

A TV Time, como mostrado na figura 6, é uma aplicação que mostra as séries vistas pelo utilizador, no entanto, encontrámos alguns problemas na navegação e alguma complexidade. Pretendemos que a nossa aplicação seja mais simples e intuitiva, e tenha também algumas funcionalidades extra, tal como a marcação dos minutos do episódio.

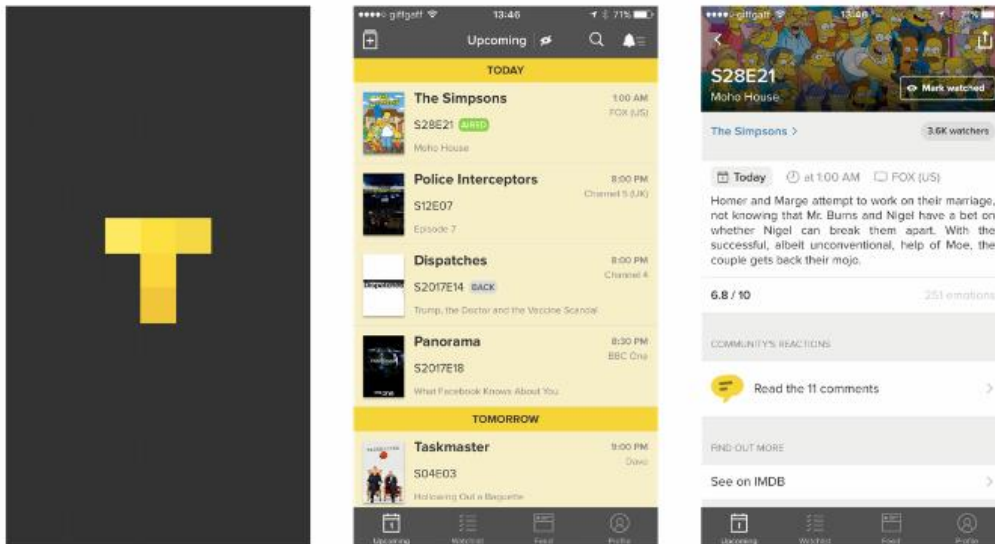


Ilustração 6 Printscreens da app TV Time

Caracterização dos utilizadores

Nesta fase imaginaram-se vários utilizadores e cenários possíveis para a aplicação, podendo desta forma perceber as funcionalidades que seriam importantes e a sua navegação.

José – tem 16 anos, gosta de ver séries de crime e comédia. Costuma ver 2 ou 3 episódios num dia e nunca deixa nenhum a meio. Vê uma série de cada vez, nunca mistura episódios diferentes. Só começa a ver as temporadas quando elas já saíram por completo. Precisa utilizar a app diariamente

Amílcar – tem 30 anos, gosta de ver séries de drama. O Amílcar é uma pessoa muito ocupada, por isso raramente consegue ver séries e quando vê, vê apenas um episódio. O Amílcar é muito esquecido, por isso precisa mesmo marcar os episódios que já viu, porque depois esquece-se e tem que estar a perder tempo a ir rever episódios.

Sofia – tem 13 anos, gosta de ver séries de drama e de thriller. A Sofia vê por volta de 1 episódio por dia, e costuma deixá-los a meio. Precisa de utilizar a app diariamente, mas por vezes esquece-se de ir lá marcar o que já viu e marca episódios que já viu com dias de atraso. Outras vezes, esquece-se por completo e deixa o episódio por marcar.

Beatriz – tem 19 anos, gosta de series de romance e aventura. A Beatriz estuda na universidade, como tem tardes livres e não tem nada para fazer fica em casa a ver séries. Vê por volta de 5 a 6 episódios por dia. A Beatriz é bastante organizada. Utiliza a aplicação varias vezes ao dia para marcar os episódios das series que vê.

Maria – tem 23 anos, gosta de terror e comédia. A Maria está à procura de emprego e quando tem tempo livre vê series. Vê temporadas e ou series inteiras numa noite ou numa tarde. Quando termina, abre logo a aplicação para marcar o que viu.

Joaquim – Tem 34 anos, adora crime, ação e aventura. O Joaquim trabalha no cinema e isso ajudou a que ao longo do tempo ganhasse uma paixão enorme não só por filmes mas também por séries. Está sempre á procura de novas séries que se assemelhem aos seus gostos para ver, e quando lhe agrada realmente determinada série quase que faz maratona e a percorre toda, vendo diariamente bastantes episódios, no entanto vai alternando com outras que lhe recomendam e por vezes esquece determinada série que estava a ver, devido a estas misturas que faz... Desta forma, a app, dar-lhe ia bastante jeito não só no aspeto de marcar onde ficou, e qual era a respetiva série que estava a ver, como ainda a marcar algumas recomendações que lhe tenham feito de uma ou outra série para que possa ver futuramente.

Cenário de utilização

A **Maria** depois de chegar a casa, liga o computador e decide ir ver uma série. Escolhe uma série nova, gosta, e vê a temporada toda. Quando acaba de ver a temporada, abre o aplicativo pela primeira vez, e clica para fazer uma conta. Na janela de Sign up, introduz o nome, o apelido, o e-mail, a password e a nacionalidade. De seguida, pesquisa pela série na barra de pesquisa, encontra a série, clica na capa e entra nos detalhes. Aí, marca logo a primeira temporada toda na checkbox que se encontra ao lado do número 1. Desta forma já não se vai esquecer que viu a 1ª temporada toda daquela série.

A **Beatriz** tem a tarde livre e por isso decide ir ver episódios que saíram recentemente. Assim que acaba de ver cada episódio, recorre logo à aplicação para marcar. Ao abrir a aplicação, uma vez que já se registou nela, aparece a "Home Page" da aplicação, já devidamente autenticada. No menu clica no icon dos vistos (caixinha com um certo) e escolhe a série desejada. Ao clicar, entra no separador da série, e lá escolhe a última temporada e faz check no último episódio. Quando acaba de marcar esse episódio, vai ver outro.

O **José** está desocupado e decide ir ver 2 episódios da série que tem estado a seguir. Ao terminar esses dois episódios, entra na aplicação (já autenticada) e vai ao separador dos favoritos. Lá, escolhe a série que está a ver e clica na capa para abrir os detalhes da série. Aí, abre o separador da segunda temporada, e faz check no episódio 3 e 4. No dia seguinte, volta à app para confirmar os episódios que já viu e continuar depois no 5º.

A **Sofia** decidiu ver um episódio duma série enquanto lanchava. Deixou o episódio a meio e esqueceu-se de ir marcar à aplicação. Nesse mesmo dia, mais tarde, a aplicação envia uma notificação que diz "Já viste uma série hoje? Aponta aqui!". Nesse momento, a Sofia recorda-se de ir marcar o episódio que tinha deixado por ver. Abre a aplicação (já autenticada), escolhe o botão de Favoritos no menu, e repara que a série que viu não estava lá. Decide ir aos Vistos. Lá encontra a série, porque já tinha marcado como visto um episódio. Aí, clica na capa da série, entra nos detalhes, clica na seta da primeira temporada, e no segundo episódio deixa uma nota que diz "10min55". E de seguida, coloca essa série nos Favoritos. Nesse dia, tinham-lhe também recomendado uma nova série de drama, por isso a Sofia decide ir procurá-la na aplicação. Vai à Homepage e na barra de pesquisa escreve o nome da série. Ao encontrá-la, nem precisa de abrir os detalhes, clica duas vezes em cima e adiciona-a rapidamente aos favoritos. Assim, da próxima vez que for utilizar a aplicação, pode ir ver consultar os minutos que ficou, e ver nos favoritos as séries que tem vistas ou por ver.

O **Joaquim**, que já usava a app, fez o seu login, introduzindo o e-mail, e a sua password. De seguida, pesquisou pela série na barra de pesquisa, encontrou a série, clicou na capa da mesma e entrou nos detalhes. Aí, o Joaquim, foi verificar em que episódio ficou e respetivos minutos na check-box. Assim, já pode retomar o episódio precisamente onde tinha ficado.

O **Amilcar**, que já era utilizador da app, estava a ver uma série, após terminar o episódio que estava a ver decidiu marcar esse mesmo episódio como visto, logo, abriu a app, fez o login, foi aos seus favoritos onde já tinha a respetiva série adicionada, no entanto entrou na check-box de outra série por engano e selecionou um episódio que não pretendia selecionar, logo entrou novamente na check-box, e retirou a seleção que tinha acabado de fazer... Fez novamente a entrada na check-box, só que desta vez na série que pretendia, e selecionou o episódio que pretendia selecionar. De seguida fez Logout.

Análise de Tarefas

Beatriz - marcar um episódio de várias séries vistas:

1. Abrir a aplicação;
2. Fazer login:
 - Colocar o email;
 - Colocar a password;
3. Ir à página da série:

- pela homepage;
- pesquisando pelo nome;
- pelos que já viu;
- pelos favoritos.

4. Selecionar o episódio pretendido:

- selecionar a temporada;
- selecionar o episódio.

5. Marcar o episódio:

- fazer check;
- colocar os minutos em que ficou.

6. Fechar a aplicação.

José – marcar vistos e favoritos

1. entrar na aplicação

- no separador clicar no icon dos favoritos
- escolher a serie que quer marcar

2. Dentro da serie

- carrega no separador da segunda temporada
- faz check no episodio 3 e 4.

3. Fecha a aplicação

4. entra na aplicação

- no separador clicar no icon dos favoritos
- escolher a serie que quer marcar

5. Dentro da serie

- carrega no separador da segunda temporada
- verifica que ja marcou os episodios

6. Fecha a aplicação

Maria – marcar uma temporada inteira

1. Abrir a aplicação;
2. Criar conta na aplicação;
 - 1.escreve o nome;
 - 2.escreve o username;
 - 3.escreve o email;
 - 4.escreve a password;
 5. escreve a nacionalidade;
 6. clica no botão de sign up.
3. Pesquisar pela serie;
 - 1.clicar na barra de pesquisa;
 - 2.escrever o nome da serie;
 - 3.clicar na imagem da serie.
- 4.Clicar no quadrado para marcar a temporada.

Terceira fase – Base de Dados

Nesta fase, fez-se o desenho da Base de Dados, no modelo relacional, para depois ser facilmente implementada no código.



Ilustração 7 Modelo Relacional da BD

A Base de Dados da aplicação vai ser constituída por 7 tabelas (User, Episódios, Séries, Vistos, Favoritos, Categorias).

A tabela "User" tem os campos (User_Name, Password, Nome, Email, Nacionalidade, Id_User).

A Tabela "Series" tem os campos (Id_Series, Nome_Serie, Id_Categoria, IMDB, Descrição).

A Tabela "Vistos" tem os campos (Id_User, Id_Episodio, visto, Minutos).

A tabela "Episodios" tem os campos (Id_Episodio, Nome_Episodio, Duração, Id_Temporada).

A tabela "Favoritos" tem os campos (Id_User, Id_Serie, Favorito).

A tabela "Categorias" tem os campos (Id_Categoria, Nome_Categoria).

Quarta Fase – Implementação do código

Ecrãs Principais da aplicação

Ecrã de Login

No ecrã do *login* (figura 8), o utilizador coloca o *email* e *password* que criou no *signup* e ao carregar no botão “Login”, entra na aplicação. Se os campos *email* e *password* estiverem vazios, o programa faz um aviso para preencher os campos todos. Se o utilizador se esquecer de preencher tanto o campo do email ou da password, a aplicação manda uma aviso para preencher o respetivo campo.



Ilustração 8 Ecrã inicial

```

1 public class MainActivity extends AppCompatActivity {
2
3     private EditText editTextEmail;
4     private EditText editTextPassword;
5
6     public static void startActivity(Context context) {
7         Intent intent = new Intent(context, MainActivity.class);
8         context.startActivity(intent);
9     }
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_main);
15
16         this.editTextEmail = findViewById(R.id.email);
17         this.editTextPassword = findViewById(R.id.password);
18
19         long userId = SessionManager.getActiveSession(this);
20         if (userId > 0) {
21             HomePageActivity.startActivity(this);
22             finish();
23         }
24     }
25
26
27     public void login(View view) {
28         String email = this.editTextEmail.getText().toString();
29         String password = this.editTextPassword.getText().toString();
30
31         if (email.isEmpty() || password.isEmpty()) {
32             Context context = getApplicationContext();
33             CharSequence text = "Preencher todos os campos!";
34             int duration = Toast.LENGTH_SHORT;
35
36             Toast toast = Toast.makeText(context, text, duration);
37             toast.show();
38
39             Log.i("MainActivity", "Preencher todos os campos!");
40             return;
41         }
42
43         User user = AppDatabase.getInstance(this).getUserDAO().getUserByEmail(email);
44         if (user != null) {
45             if (user.getPassword().equals(password)) {
46                 HomePageActivity.startActivity(this);
47                 SessionManager.saveSession(this, user.getUserId());
48                 finish();
49             } else {
50                 Context context = getApplicationContext();
51                 CharSequence text = "Password Errada!";
52                 int duration = Toast.LENGTH_SHORT;
53
54                 Toast toast = Toast.makeText(context, text, duration);
55                 toast.show();
56
57                 Log.i("MainActivity", "Password Errada!");
58             }
59         } else {
60             Context context = getApplicationContext();
61             CharSequence text = "Utilizador Não Existe!";
62             int duration = Toast.LENGTH_SHORT;
63
64             Toast toast = Toast.makeText(context, text, duration);
65             toast.show();
66
67             Log.i("MainActivity", "Utilizador Não Existe!");
68         }
69     }
70
71     public void signup(View view) {
72         SignUpActivity.startActivity(this);
73     }
74 }

```

Ilustração 9 Código da MainActivity

Na função "onCreate" depois de verificar que o user existe e ter a secção iniciada a class "HomePageActivity" é inicializada.

Na função "Login" o que o utilizador colocar nos campos, email e password, vai ser verificado. Caso tudo esteja de acordo com a base de dados o Log In é feito com sucesso, caso o utilizador não preencha nenhum campo a aplicação avisa o utilizador com a mensagem "Preencher todos os campos!", se os campos email ou password não estiverem preenchidos a aplicação retorna uma mensagem de erro com o respetivo campo errado ou não preenchido.

Ecrã de Signup

No ecrã "Sign Up" o utilizador para criar uma conta escreve nos campos designados e carrega no botão "Sign Up".

Caso qualquer um dos campos não esteja preenchido, o utilizador receberá uma mensagem de aviso com o respetivo campo não preenchido.

The image shows a mobile application screen titled "Sign Up" with a close button (X) in the top right corner. The screen has a light gray background. There are three input fields with labels "Email", "Password", and "Nacionalidade" stacked vertically. Each field has a horizontal line below the label. At the bottom of the screen, there is a blue button with the text "Sign Up" in white.

Ilustração 10 Ecrã Sign Up

```
1 public class SignUpActivity extends AppCompatActivity {
2
3     private EditText editTextEmail;
4     private EditText editTextPassword;
5     private EditText editTextNacionalidade;
6
7
8     public static void startActivity(Context context) {
9         Intent intent = new Intent(context, SignUpActivity.class);
10        context.startActivity(intent);
11    }
12
13    @Override
14    protected void onCreate(Bundle savedInstanceState) {
15        super.onCreate(savedInstanceState);
16        setContentView(R.layout.activity_sign_up);
17
18        this.editTextEmail = findViewById(R.id.email);
19        this.editTextPassword = findViewById(R.id.password);
20        this.editTextNacionalidade = findViewById(R.id.nacionalidade);
21    }
22
23    public void signUp(View view) {
24        String email = this.editTextEmail.getText().toString();
25        String password = this.editTextPassword.getText().toString();
26        String nacionalidade = this.editTextNacionalidade.getText().toString();
27
28        if (email.isEmpty() || password.isEmpty() || nacionalidade.isEmpty()) {
29
30            Context context = getApplicationContext();
31            CharSequence text = "Tem de preencher os campos todos!";
32            int duration = Toast.LENGTH_SHORT;
33
34            Toast toast = Toast.makeText(context, text, duration);
35            toast.show();
36
37            Log.i("SignUpActivity", "Tem de preencher os campos todos");
38            return;
39        }
40
41        HomePageActivity.startActivity(this);
42        User user = new User(1, email, password, nacionalidade);
43        AppDatabase.getInstance(this).getUserDAO().insert(user);
44        finish();
45    }
46
47
48    public void gologin (View view) {
49        finish();
50    }
```

Ilustração 11 Código da SignUpActivity

Na função "SignUp" se nenhum dos campos estiverem preenchidos a aplicação manda uma mensagem de aviso para preencher todos os campos. Se for o caso de só um dos campos não tiver preenchido a aplicação vai mandar uma mensagem de aviso para o utilizador preencher o respetivo campo.

Se todos os campos estiverem preenchidos a aplicação vai abrir o "HomePageActivity", guardar o utilizador criado na Base de Dados e vai fechar a "SignUpActivity".

Na função "GoLogin" assim que se faz o click do icon "X" a "ActivitySignUp" fecha e a "MainActivity" é inicializada.

Ecrã da Página Inicial

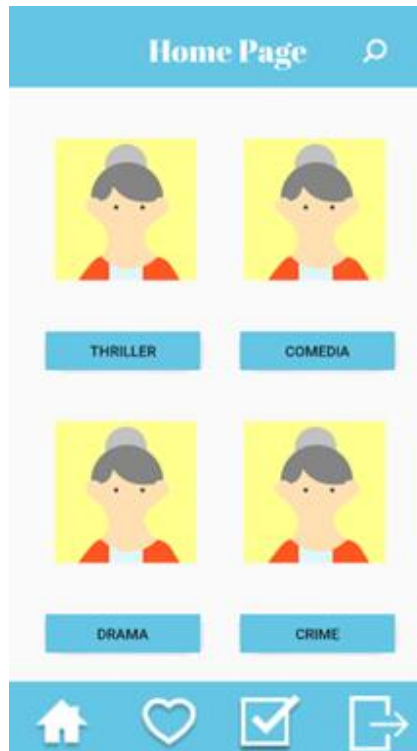


Ilustração 12 Ecrã Home Page

```

1- public class HomePageActivity extends AppCompatActivity {
2-
3-     private ImageView imageViewThriller;
4-     private ImageView imageViewComedia;
5-     private ImageView imageViewDrama;
6-     private ImageView imageViewCrime;
7-
8-     public static void startActivity(Context context) {
9-         Intent intent = new Intent(context, HomePageActivity.class);
10-        context.startActivity(intent);
11-    }
12-
13-    @Override
14-    protected void onCreate(Bundle savedInstanceState) {
15-        super.onCreate(savedInstanceState);
16-        setContentView(R.layout.activity_home);
17-        this.imageViewThriller = findViewById(R.id.image1);
18-        this.imageViewComedia = findViewById(R.id.image2);
19-        this.imageViewDrama = findViewById(R.id.image3);
20-        this.imageViewCrime = findViewById(R.id.image4);
21-
22-        this.populateImageView();
23-    }
24-
25-    private void populateImageView() {
26-        SeriesDAO seriesDAO = AppDatabase.getInstance(this).getSeriesDAO();
27-        List<Serie> seriesThriller = seriesDAO.getSeriesByCategory(Categoria.THRILLER);
28-        List<Serie> seriesComedia = seriesDAO.getSeriesByCategory(Categoria.COMEDIA);
29-        List<Serie> seriesDrama = seriesDAO.getSeriesByCategory(Categoria.DRAMA);
30-        List<Serie> seriesCrime = seriesDAO.getSeriesByCategory(Categoria.CRIME);
31-
32-        if (seriesThriller.size() > 0) {
33-            Serie serieThriller = seriesThriller.get(0);
34-            if (serieThriller != null) {
35-                Glide.with(this).load(serieThriller.getImagem()).into(this.imageViewThriller);
36-            }
37-        }
38-
39-        if (seriesComedia.size() > 0) {
40-            Serie serieComedia = seriesComedia.get(0);
41-            if (serieComedia != null) {
42-                Glide.with(this).load(serieComedia.getImagem()).into(this.imageViewComedia);
43-            }
44-        }
45-
46-        if (seriesDrama.size() > 0) {
47-            Serie serieDrama = seriesDrama.get(0);
48-            if (serieDrama != null) {
49-                Glide.with(this).load(serieDrama.getImagem()).into(this.imageViewDrama);
50-            }
51-        }
52-
53-        if (seriesCrime.size() > 0) {
54-            Serie serieCrime = seriesCrime.get(0);
55-            if (serieCrime != null) {
56-                Glide.with(this).load(serieCrime.getImagem()).into(this.imageViewCrime);
57-            }
58-        }
59-    }
60-
61-    public void gofavoritos(View view) {
62-        FavoritosActivity.startActivity(this);
63-    }
64-
65-    public void govistos(View view) {
66-        VistosActivity.startActivity(this);
67-    }
68-
69-    public void logout(View view) {
70-        SessionManager.deleteSession(this);
71-        MainActivity.startActivity(this);
72-        finish();
73-    }
74-
75-    }
76-
77-    }
78-
79-    }

```

Ilustração 13 Código da HomePageActivity

Na função "PopulateImageView" as categorias das imagens são verificadas e selecionadas para as suas respetivas categorias, as imagens das series guardadas na base de dados de acordo com a categoria vão ser inseridas nos respetivos lugares.

Na função "GoFavoritos" a "FavoritosActivity" é inicializada.

Na função "GoVistos" a "VistosActivity" é inicializada.

Na função "LogOut" elimina a sessão do "SessionManager", inicializa a "MainActivity" e fecha a activity "HomePage".

Implementação da BD

Para implementar a BD, utilizámos a biblioteca Room, uma ferramenta que ajuda à implementação de bases de dados no Android Studio. Na figura abaixo encontra-se o código utilizado para inserir dados na BD. Para inserir imagens na BD utilizou-se o Glide, uma biblioteca que permite facilmente inserir imagens através do *url*. (ilustração 18).

```

1 public abstract class AppDatabase extends RoomDatabase {
2
3     public abstract UserDao getUserDAO();
4     public abstract SeriesDAO getSeriesDAO();
5
6     private static AppDatabase INSTANCE;
7
8     public static AppDatabase getInstance(Context context){
9         if (INSTANCE == null) {
10             INSTANCE = Room.databaseBuilder(
11                 context.getApplicationContext(),
12                 AppDatabase.class,
13                 "UserDB")
14                 .allowMainThreadQueries()
15                 // inserir coisas na base de dados
16                 .addCallback(new Callback() {
17                     @Override
18                     public void onCreate(@NonNull SupportSQLiteDatabase db) {
19                         super.onCreate(db);
20                         //Meter codigo sql
21                         db.execSQL("INSERT INTO Serie(nomeSerie, idCategoria, Imdb, descricao, imagem) " +
22                             "VALUES ('Backlist', 4, '8,1',' +
23                             \"Uma nova criadora de perfis do FBI, Elizabeth Keen, teve toda a sua vida arrancada quando um criminoso misterioso, Raymond Reddington, que escapou da captura por décadas, s
24
25                         db.execSQL("INSERT INTO Serie(nomeSerie, idCategoria, Imdb, descricao, imagem) " +
26                             "VALUES ('Riverdale', 3, '7,3',' +
27                             \"After the death of one of the rich and popular Blossom twins on the 4th of July, the small town of Riverdale investigates the murder. The series starts in September, the br
28
29                         db.execSQL("INSERT INTO Serie(nomeSerie, idCategoria, Imdb, descricao, imagem) " +
30                             "VALUES ('American Dad', 2, '7,4',' +
31                             \"Stan Smith, who works for the C.I.A. and is constantly on the alert for terrorist activity, will go to extremes to protect his beloved America from harm; as evidenced by th
32
33                         db.execSQL("INSERT INTO Serie(nomeSerie, idCategoria, Imdb, descricao, imagem) " +
34                             "VALUES ('The Walking Dead', 1, '8,3',' +
35                             \"Sheriff Deputy Rick Grimes gets shot and falls into a coma. When awoken he finds himself in a Zombie Apocalypse. Not knowing what to do he sets out to find his family, afte
36
37                         db.execSQL("INSERT INTO Serie(nomeSerie, idCategoria, Imdb, descricao, imagem) " +
38                             "VALUES ('Game of Thrones', 3, '9,4',' +
39                             \"In the mythical continent of Westeros, several powerful families fight for control of the Seven Kingdoms. As conflict erupts in the kingdoms of men, an ancient enemy rises
40
41                         db.execSQL("INSERT INTO Serie(nomeSerie, idCategoria, Imdb, descricao, imagem) " +
42                             "VALUES ('The Chilling Adventures of Sabrina', 3, '7,7',' +
43                             \"Reimagines the origin and adventures of Sabrina: the Teenage Witch as a dark coming-of-age story that traffics in horror, the occult and, of course, witchcraft. Tonally in
44
45                         db.execSQL("INSERT INTO User(password, email, nacionalidade)"+ "VALUES('123','teste','Potuguesa')");
46                         db.execSQL("INSERT INTO User(password, email, nacionalidade)"+ "VALUES('123','123','123')");
47
48                         db.execSQL("INSERT INTO Visto (idUser, idSerie, visto)"+ "VALUES(2, 6, 'true')");
49
50                     }
51                 })
52             }
53         }
54     }
55 }

```

Ilustração 14 Código da AppDatabase


```
1 @Dao
2 public interface UserDao {
3
4     @Query("SELECT * From User")
5     public List<User> GetAllUsers();
6
7     @Update
8     public void update (User User);
9
10    @Insert
11    public void insert(User User);
12
13    @Insert
14    public void insertAll(List<User> InsertAll);
15
16    @Query("SELECT * FROM User WHERE email = :email")
17    User getUserByEmail(String email);
18
19 }
20
```

Ilustração 15 DAOs da classe User

```
1 @Dao
2 public interface SeriesDAO {
3
4     @Query("SELECT * FROM Serie")
5     List<Serie> getAllSeries();
6
7     @Query("SELECT * FROM Serie WHERE idCategoria = :id")
8     List<Serie> getSeriesByCategory(long id);
9
10
11 }
```

Ilustração 16 DAOs da classe Series

```
1 @Dao
2 public interface CategoriasDAO {
3
4     @Query(" SELECT * FROM Categoria")
5     List<Categoria> getAllCategorias();
6
7 }
8
```

Ilustração 17 DAOs da classe Categorias

Os DAOs que vemos nas classes acima são as *queries* SQL utilizadas para mostrar dados da BD na aplicação.

```

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:23.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.android.support:design:23.0.0'
    testImplementation 'junit:junit4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:1.1.0'

    // Glide
    implementation 'com.github.bumptech.glide:glide:3.8.0'
    annotationProcessor 'com.github.bumptech.glide:compiler:3.8.0'

    // Room
    def room_version = "1.1.1"
    implementation "android.arch.persistence.room:runtime:$room_version"
    annotationProcessor "android.arch.persistence.room:compiler:$room_version"
    // For Kotlin use kapt instead of annotationProcessor
}

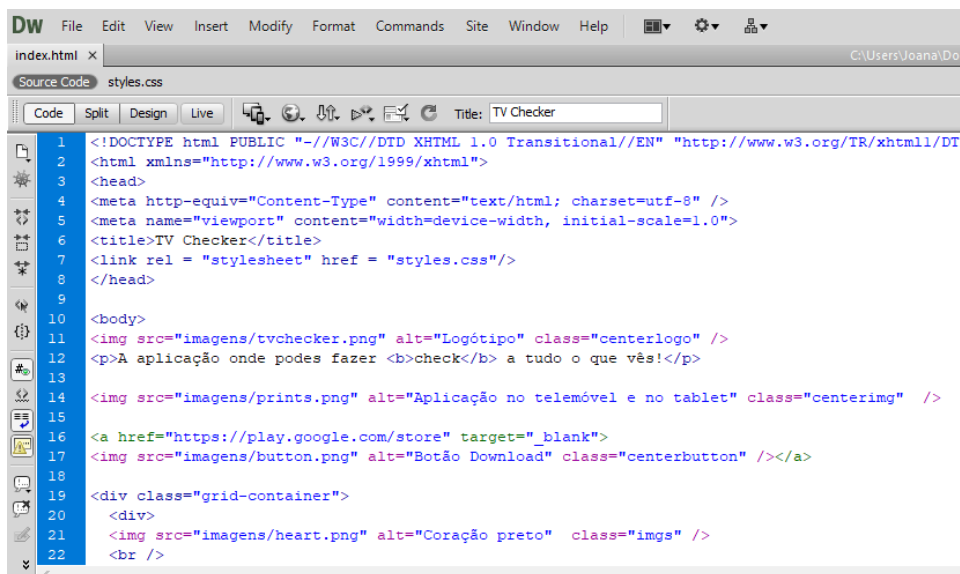
```

Ilustração 18 Código do build.gradle (Glide e Room)

Website de divulgação

Elaborou-se um *website* de divulgação da aplicação, com o objetivo de mostrar aos utilizadores de que se trata a mesma e onde podem fazer o respetivo *download*. Neste *site* são descritas todas as funcionalidades da aplicação e são mostrados os ecrãs principais.

O *site* foi criado de raiz em HTML e CSS, utilizando o *software* da Adobe, Dreamweaver.

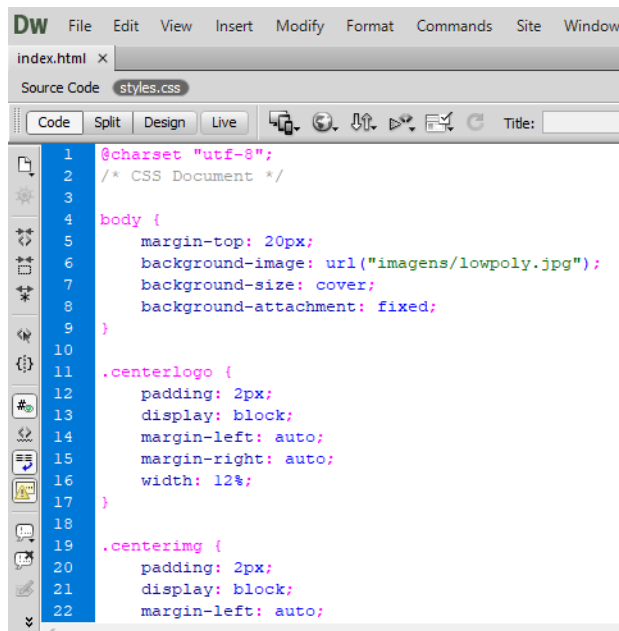


```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD:
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>TV Checker</title>
7 <link rel = "stylesheet" href = "styles.css"/>
8 </head>
9
10 <body>
11 
12 <p>A aplicação onde podes fazer <b>check</b> a tudo o que vês!</p>
13
14 
15
16 <a href="https://play.google.com/store" target="_blank">
17 </a>
18
19 <div class="grid-container">
20 <div>
21 
22 <br />

```

Ilustração 19 HTML



```

1 @charset "utf-8";
2 /* CSS Document */
3
4 body {
5     margin-top: 20px;
6     background-image: url("imagens/lowpoly.jpg");
7     background-size: cover;
8     background-attachment: fixed;
9 }
10
11 .centerlogo {
12     padding: 2px;
13     display: block;
14     margin-left: auto;
15     margin-right: auto;
16     width: 12%;
17 }
18
19 .centerimg {
20     padding: 2px;
21     display: block;
22     margin-left: auto;

```

Ilustração 20 CSS

O *website* ficou bastante fiél aos desenhos iniciais e ao protótipo realizado em Adobe Illustrator. O fundo manteve-se, aproveitaram-se as imagens, logótipos e botões. A disposição dos *printscreens* dos ecrãs alterou-se ligeiramente (Figura 21), porque durante a criação, reparou-se que adaptado ao telemóvel ficaria menos apelativo, teria que ficar uma imagem por cima da outra e não lado a lado, ficando muito espaço vazio dos lados. Sendo assim, preferiu-se colocar, numa resolução grande, os quatro ecrãs lado a lado, e, em resoluções mais pequenas, organizados verticalmente.



Ilustração 21 Design no Ai

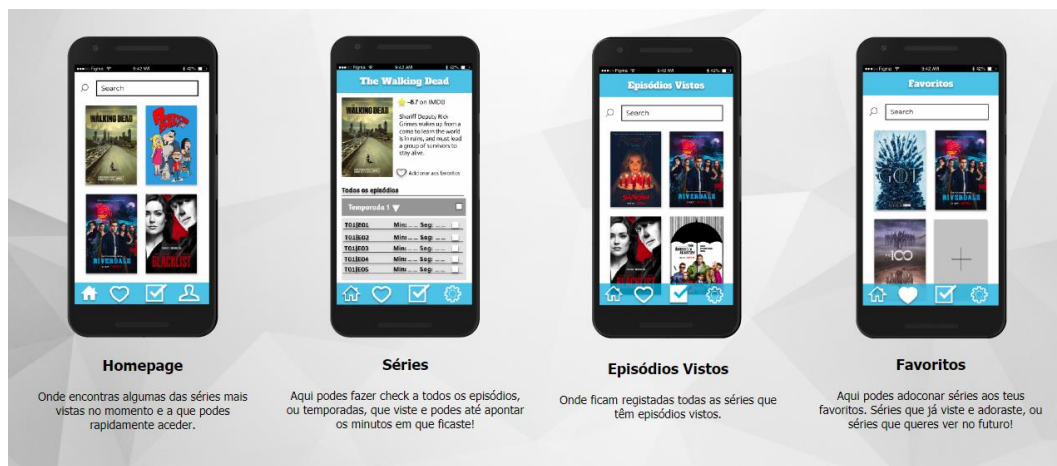


Ilustração 22 Website implementado

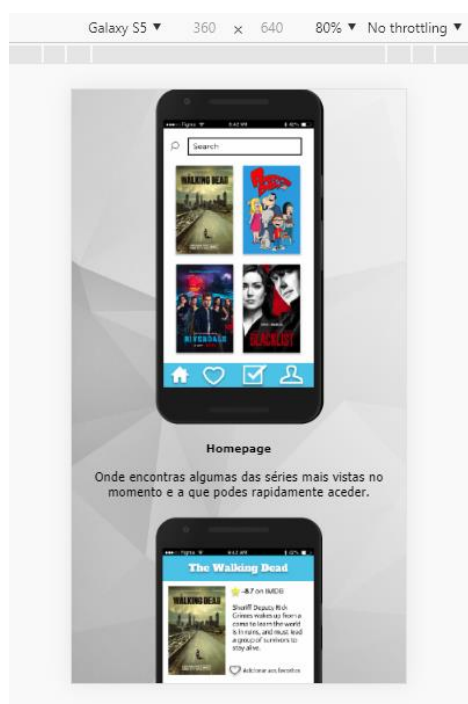


Ilustração 23 Website implementado visão telemóvel

No sítio *web* temos também um botão “Download para Android” que nos remete para uma página a Play Store, numa página à parte. É sempre de maior interesse colocar todos os links externos a remeterem para uma página à parte, para que o utilizador mantenha sempre o *site* aberto.

Todos os estilos foram colocados no ficheiro “styles.css”.

HTML e CSS

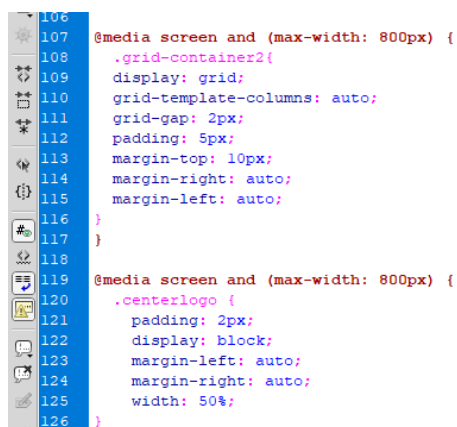
O “body” tem a imagem definida e o fundo da página fixo.

Todas as imagens e botões estão numa classe específica no CSS, para poder ser designado o seu próprio tamanho e espaçamento, dessa forma sendo também mais fácil definir os mesmos para ecrãs mais pequenos.

Os três ícones a preto têm todas a mesma dimensão e estão na disposição de *grid*, para que pudessem estar lado a lado e com uma certa distância pretendida.

Utilizou-se várias vezes o “margin-left: auto; margin-right: auto;” para definir o alinhamento central das imagens.

Utilizaram-se também dois tipos de parágrafos, classes “p” e “p.smaller”, para se poder ter textos maiores e mais pequenos.



```
106
107 @media screen and (max-width: 800px) {
108   .grid-container2{
109     display: grid;
110     grid-template-columns: auto;
111     grid-gap: 2px;
112     padding: 5px;
113     margin-top: 10px;
114     margin-right: auto;
115     margin-left: auto;
116   }
117 }
118
119 @media screen and (max-width: 800px) {
120   .centerlogo {
121     padding: 2px;
122     display: block;
123     margin-left: auto;
124     margin-right: auto;
125     width: 50%;
126   }
```

Ilustração 24 CSS adaptado a ecrãs

Na figura acima, verifica-se a forma como se adaptaram os elemtnos para ecrãs mais pequenos, definindo que se o tamanho máximo fosse de 800px, os elementos teriam outras definições.

Conclusões

Com este trabalho aprendeu-se bastante sobre como incluir Bases de Dados em aplicações para Android, imagens, *logins*, e várias funcionalidades bastante utilizadas nas aplicações atualmente. Incluiu também a produção de um *website* para publicitar a aplicação e os seus objetivos, de modo a ter presença na *internet* e explicar aos utilizadores para que serve a aplicação e porque devem utilizá-la. A divulgação *online* é uma parte inicial de qualquer projeto e considera-se importante que o projeto tenha incluído esta vertente.

Este projeto foi também bastante enriquecedor na medida em que analisámos todas as possíveis utilizações da aplicação, caracterizando dessa forma o público-alvo e analisando posteriormente todas as funcionalidades necessárias a cada tipo de utilização possível, desta forma percebendo com facilidade o que esta aplicação deveria ter e os seus objetivos.

Passámos por diversas fases que ajudam a organizar e definir propriamente projetos deste tipo, e é bastante importante que se tenha em mente que este tipo de trabalhos não é possível sem passar por várias fases de análise, desenho, e objetivos bases do projeto, passando mais tarde para a implementação.

Este tipo de projetos são sempre desafios enormes para os alunos, dado que é necessário uma grande organização de grupo, com uma gestão de tarefas apropriada. Propôs-se um trabalho complexo de início, e para algumas funcionalidades não se conseguiu apresentar uma conclusão, tendo assim em conta que o projeto tem muitas áreas de melhorias.

Futuramente, pretende-se terminar as funcionalidades em falta e apresentar uma aplicação mais completa para o utilizador.